POSIX – Portable Operating System Interface -- Consistency Standard

Unix

Darwin

GNU/Linux

Solaris

BSD

OS10 very popular with Web developers

Windows – Not POSIX at all – less desirable choice for Web development

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

$ man ls -- manual for ls command

$ exit -- exit BASH end session

$ ls (list) List Files [ command name -- the verb]

Output: documents hello.txt -- folder: documents text file: hello.txt

$ ls –l (long form) -- ls[space] –l [the option – modify behavior of command - like an adverb]

total 8 total number of files

drwxrwxr-x treehouse treehouse (user and group name treehouse) 4096 (size bytes ) date of file and name

directory read write execute – three rwx triplets

first triplet rwx - user who created it

second triplet rwx – group – can be multiple users in a group

third triplet r-x – public permission for everyone else – read and execute

file with .in front of name – hidden file

will not show up in ls listing

$ ls –a to see hidden files (usually config files and not to worry – dot files)

[argument – what we want our command to operate on]

$ ls /etc show all files in etc directory (pass /etc argument) name of folder on system

$ls –l /etc

root -- user is root – administrator user – default all powerful user

$ clear – clear backlog of screen and prompt to top

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Move around the File System**

$ pwd -- print working directory [command] /home/treehouse -- where currently are in file system

Home directory:

different home directory for each user on the computer

Linux: /home/username

MAC: /users/username

Shortcut name: ~ “my home directory” abbreviated path. ~ means you are at home

$ cd – Change Directory

TAB completion – to fill in things for us ex. cd do –TAB to fill in rest – TAB through with multiple presses

History – up arrow to give the last command

~/documents $ -- home directory and inside documents directory

Relative path: cd name or cd ..

Absolute path: starts with a point: **cd /home/treehouse/documents** OR **cd ~/documents**

Absolute path begins with a slash / meaning ls

**ROOT**

To home directory: cd ~

-bash – (bash is our shell – most popular) others are ZSH CSH TCSH ending with sh

Program we interact with – shell

Program taking input and output - can do color and TAB and other stuff too

**Go down a directory:**

$ cd documents -- no / slashes separate multiple levels of directories that we want to move through

prompts gives context where we are: treehouse ~/documents $

**Go up a directory:**

$ cd ..

$ cd ../.. go up two directories

**How to go HOME:**

$ cd ~

$ cd

$ cd /home/treehouse

$ cd /home/$USER

$ cd $HOME

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Reading Files**

Program: less -- view content of hello.txt -- PAGER shows one page at a time

less hello.txt argument = file you want to view

the less program will take up the entire console screen

q Quits out of the less command

Program: more is the old version of the program

less is better

Program: cat concatenate files – also print out file

Concatenate files:

$ cat hello.txt documents/how\_to\_go\_home.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Editing Files**

Program: nano text editor (free)

Program: pico text editor

Program: Vim

Program: Emacs

Controls: ^ represents control key

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Move Coy Delete Files**

Move:

$ mv takes two arguments: from where we want to move to where we want to move

Rename from hello.txt to hi.txt

$ mv hello.txt hi.txt equivalent to renaming it

Move hi.txt into the documents directory (from current directory) with same file name:

$ mv hi.txt documents/

. represents for moves

$ mv documents/hi.txt . the “.” Is the second argument

Move and rename (from hi.txt to hello.txt):

$ mv hi.txt documents/hello.txt

**Move works for directories too:**

Move and rename documents to docs

$ mv documents docs

Copy Command:

$ cp – like mv but leaves the original one in place

$ cp hello.txt hi.txt -- make another copy with a new name

Copy directory: can’t just copy a directory. Need to do recursively.

$ cp –r documents docs - pass a special option called –r -- copy everything recursively

For directory and all files within it and all sub-directories

**Remove file** command:

$ rm -- PERMANENT remove

**Remove directory:**

$ rm –r -- pass the –r flag

Make a directory:

$ mkdir and pass name (pictures)

$ mkdir pictures

Create directory inside directory:

$ mkdir documents/notes

$ mkdir documents/notes/console

Nested directory trick:

$ mkdir –p – create any level deep directory even if it does not exist yet

$ mkdir –p documents/notes/console/part1 -- creates all levels deep

Create File:

$ touch name

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Create New User**

$ whoami -- displays the user you are currently logged in as

$ sudo adduser daniel

**Switch User** command: log in as a different user

$ su daniel

$ exit - exit from the switched user

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**File Permissions rwx**

Read

Write – update or delete

Execute – only comes into play with programs

Who? User (and group owner), Group, Other Users ugo == rwxrwxrwx

u g o : Who

**drwxrwxrwx (d is for directory) user (treehouse) group (treehouse)**

$ chmod -- changes the permissions (mode) of file or directory -- change mode of file

$ chmod user/group/other

$ chmod o+w (add write permission to other) hello.txt (“-“S to remove a permission)

$ chmod +x (update for everyone no need to add a who)

Decimal

0 1 2 3 4 5 6 7 8 9

10 11 12 13 14 15 16 17 18 19

Octal

0 1 2 3 4 5 6 7

10 11 12 13 14 15 16 17

7 r w x

6 r w -

5 r - x

4 r - -

3 - w x

2 - w -

1 - - x

0 - - -

$ chmod 777 hello.txt ----> drwxrwxrwx

$ chmod 640 hello.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**File Ownership**

$ chown -- changes the owner of a file or directory

$ sudo chown daniel hello.txt -- change user ownership

$ sudo chown daniel:daniel hello.txt -- change user and group ownership

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Sudo Ownership**

$ sudo -- run a command as a super user -- run a command with privileges of root without explicitly switching to the root user or neleding to know the root password

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Previous Command shortcut:**

$ !!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Processes: Core component of POSIX operating systems**

Nano is a program

Executed nano is a process – process is an instance of the program

Can have many instances of nano – several processes

Explore processes on our machine:

$ top

**“q”**  to leave

$ ps – program that lists out processes

ex. BASH – program interpreting our keystrokes and formatting output to our screen

$ ps aux -- view all processes for the user x (all users) – all processes on your machine

Find process ID for a specific task

$ ps aux | grep “top” -- to find a specific process ID

Grep is a tool that allows us to take some input, search for a certain pattern and filter only the lines with that pattern on it

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Pausing and Resuming and Killing**

Create new file to edit: $ nano demo.txt

Pause this job (process that belongs to me): ctrl +z

Return: $ fg -- foreground or $ fg 2 (the number of the job)

$ jobs -- list out all the jobs -- List of processes in current session – jobs are for A Session

$ top & -- start job and automatically put it in the back

Kill processes:

Signals – specify different behavior – standardized

Signal: TERM

Ctrl +c -- sends TERM signal -- clean up and exit

$ kill 1450 -- kill process 1450 -- sends TERM or terminate signal to the process

$ kill –TERM 1456 this is the normal terminate signal

If process can’t see a kill signal:

Last Resort: Turns process off immediately – and nothing the process can do

$ kill –KILL 1456

$ kill –SIGKILL 1456

$ kill –9 1456

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Environment Variables**

Environment variables written in all UPPER CASE and values they hold are strings

$ env -- print out all environment variables

PS1=\u \w $ defines what the prompt is \u = username \w – present working dir

HOME=/home/treehouse

PATH=/usr/local/sbin:

$ echo -- echo back argument you give it

$ echo $HOME 🡪 /home/treehouse echo receives the value of the variable

$ cd $HOME with use the value of the variable and will go to /home/treehouse

Change a variable:

$ PS1=”\w >” -- this will change the prompt

$ bash -- start new bash session -- start a new instance of bash -- child of original one

$ exit – stop the instance of bash currently in. take to original one

Create environment variable:

$ MESSAGE=”Hello World” only good in the bash instance created – stays in own process – not passed down

EXPORTING -- $MESSAGE is LOCAL env variable, only visible to us

run another program – only have access to env variable that are exported from session that created it

$ export MESSAGE=”Hi World” export to any processes we run

PATH

$ echo $PATH – list of directories separated by colons

$ which echo -- show directory program is in

$ which nano

Update path:

$ export PATH=/home/treehouse/bin:$PATH -- Giving PATH a new value

Now need to place this in a file that will run (on startup):

$ ls –la -- show file called **.bashrc** -- file that starts every time we startup

-a flag shows all files including hidden

$ nano .bashrc -- to edit the file

Add to this file:

export PS1=”\u \w \$”

export PATH=/home/treehouse/bin:$PATH

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Find and Grep -- find files and text**

Locate files base on file name.

“.” Start search from current directory. Pass “–name” argument and name of file

$ find . –name “how\_to\_go\_home.txt”

Search through entire system – start at root “/”

$ find / -name “sudoer”

Search specific directories:

$ find documents bin –name “how\_to\_go\_home.txt” search “documents” and “bin” directories

Search inside a file:

Grep: grep pattern filename

Program/utility: grep -- Global Regular Expression Print -- globally search through a file for a regular expression (pattern that defines a particular expression) – looks through file for a pattern

$ grep –n -i “is” hello.txt

pass –n option for line number pass the pattern “is” pass files or list of files

pass –i for case insensitivity

pass –v for lines without a patter -- invert. Find lines that do not contain some patter

$ man grep to learn more

Used grep with processes:

$ ps aux | grep “init”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Pipes and Redirection**

Standard In

Standard Out

Standard Error -- by default also prints out to terminal

Default for the standard In is the keyboard input

If we omit the file name from a tool that can read from standard input

We can then use our keyboard to input our text that we want to search for a pattern in

$ grep hello

Type in: Hello world << this is the input

Output: Hello << pattern found

Cntrl +d – signals end of input

Works with other commands like $ echo (if don’t pass anything)

Redirection -- standard In

$ grep this < FILE redirect a file as INPUT instead of the KEYBOARD standard input

$ grep this < hello.txt SAME AS: $ grep this hello.txt

Redirection -- standard Out – standard out is the terminal

$ grep this > FILE

$ grep this hello.txt > hello\_grep-txt (the file created and filled with the output of running the command)

Append output and not overwrite:

$ grep line hello.txt >> hello\_grep.txt >> APPEND

**Redirect Standard Error Output (STREAM #2)** to a file called “error log”

$ find / -name “sudoers” 2> error\_log.txt -- All errors go into a file

Garbage Can File to get rid of unwanted data: /dev/null -- deletes everything written to id

$ find / -name “sudoers” 2> /dev/null -- All error are deleted

**Pipe Command A | Command B**

Pipe the output of Command A to the input of Command B

$ ps aux | grep bash OUTPUT from all processes becomes INPUT for grep and searching for bash

**Program: sort** sorts lines of standard input and send it to standard output (default by string comparison)

$ ps aux | grep bash | sort

Send to a file:

$ ps aux | grep bash | sort > sorted\_bash\_procs.txt

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Building and install Software From Source**

**General Process:**

1 download source file

2 untar the file

3 Run ./configure script

4 Run make command

5 Run sudo make install

Manual way of building and installing SW on Linux based machine

Commands

* sudo apt-get update  Update your computer's catalog of available software
* sudo apt-get install build-essential Install the tools needed to build software from source code
* curl -O URL  Download the file at the URL
* tar -xvf FILENAME.tar.gz  Decompress the tar.gz file to the current directory
* ./configure  Run the configure script that comes with the source code. This creates a Makefile
* make  Run the build specified in the Makefile
* sudo make install  Run the install script from the Makefile. This installs the program

DEMO: Install SQLite (lightweight, file-based database) -- Install from Source

**Install a couple tools on machine** – only need to do this once (for UBUNTU Linux)

NEED **build-essential** package to be able to build programs from source

$ sudo apt-get update -- part of the package manager system

Now we can install programs

$ sudo apt-get install build-essential (package to install is: “**build-essential**”)

Installs requested tools

$ which make Confirm tools installed

which - command which takes name of program and tell where it lives on system

make - program used for building things -- tools we use to install programs from source

Now need source code for SQLite -- Source Code or tarball tar file. Tar.gz is much like a .zip file

**Download Source File**

Copy the link address –

$ curl –O URL curl – program used for making request from internet

-O will save the file we paste in to a file on our machine

Right click on console and “paste” or edit paste

$ ls to confirm downloaded to home directory

$ mkdir src

$ mv sqlite-autoconf-307.tar.gz src/

$ cd ~/src

**Untar the file:**

$ tar –xvf sqlite-autoconf-307.tar.gz

-xvf x: extract v: verbose output to see all that happens f: pointing to tar file to extract

Creates a new directory – go into:

$ cd sqlite-autofonc-307/

See a bunch of files

Steps to install a program from source

**Step 1:** run program called configure that exists inside the project directory

$ ./configure ./ run a program right here in our current directory

This script looks through our system and prepare more config files to build the program

This script created a Makefile (file that specifies how to build the program) compatible with our system

**Step2:** Execute the Makefile using the program: make (which is what we installed with build-essential)

From with same directory as Makefile (project directory itself):

$ make building program from all source files

**Step 3:** Program is built but not yet installed. Install the program

Executables built and everything that it needs

However they’re still all in the src/sqlite directory

Need the executables to be somewhere in our PATH

That way we can just run the executable like sqlite3

$ sudo make install

move everything that it has to somewhere that will be accessible in our path. Prepend with sudo to make sure when it tries to move the files into their final location, we have permission to do so

**Confirm sqlite was installed**

$ which sqlite3

/usr/local/bin/sqlite3 /usr/local/bin is part of our PATH

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction to Package Managers**

**Install git or servers like Apache, databases MySQL or even SQLite**

Commands

* apt-get update Update your package catalog on your computer
* apt-cache search PATTERN Search the available packages for a pattern
* apt-get install PACKAGE Install one or more packages
* apt-get upgrade Upgrade to the latest version of all the packages installed
* apt-get remove PACKAGE Remove or uninstall package from your computer

Package manager for UBUNTU LINUX is called **APT (Advanced Packaging Tool)**

UBUNTU is a member of a family of Linux distributions that are based on Debian Linux

APT is used for Debian-based Linuxes and its related distros

Others: Pacman, RPM and YUM on the various other distros

Mac OS X has its own package manager called Homebrew

Instead of installing a program from source. Instead use a package manager

First thing to do:

$ sudo apt-get update

Update our local databases of the available packages. Run every once in a while. Make sure our database has the complete list of packages up to date including the most up to date versions

**Install the program Git:**

$ sudo apt-get install git

Or search for things using the APT system: Program: APT-Cache

$ apt-cache search git search for packages based on a pattern – packages with word git

Package we are looking for: git

$ sudo apt-get install git

$ which git

/usr/bin/git

Periodically **update** our packages: updates and all new security patches

$ sudo apt-get upgrade

**By default will upgrade the packages that it knows about**

Run upgrade after update. Update the system to be aware of new packages

Uninstall a program:

$ sudo apt-get remove git only removes git

Unistall dependent programs:

$ sudo apt-get autoremove